**AI Lab Submission**



Session: 2022 – 2026

**Submitted by:**

Muhammad Tayyab 2022-CS-133

**Submitted to:**

Mr. Nauman Shafee

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

# Case Study 1: Customer Purchasing Behavior Analysis

## Task 1:

## count = 1000

## customerID = np.arange(1, count + 1)

## age = np.random.randint(18, 71, count)

## annualIncome = np.random.randint(20000, 120001, count)

## gender = np.random.choice(['Male', 'Female'], count)

## purchased = np.random.choice([0, 1], count)

## customersData = pd.DataFrame({

## 'CustomerID': customerID,

## 'Age': age,

## 'AnnualIncome': annualIncome,

## 'Gender': gender,

## 'Purchased': purchased

## })

## Task 2:

first10Rows = customersData.head(10)

missingValues = customersData.isnull().sum()

## Task 3:

medianIncome = customersData['AnnualIncome'].median()

customersData['AnnualIncome'] = customersData['AnnualIncome'].fillna(medianIncome)

## Task 4:

labelEncoderGender = LabelEncoder()

customersData['Gender'] = labelEncoderGender.fit\_transform(customersData['Gender'])

## Task 5:

scaler = MinMaxScaler()

customersData[['Age', 'AnnualIncome']] = scaler.fit\_transform(customersData[['Age', 'AnnualIncome']])

## Task 6:

# Histogram of Age

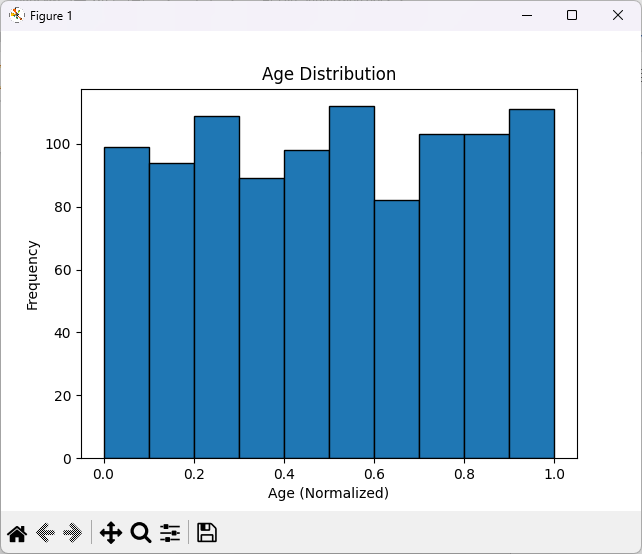
plt.hist(customersData['Age'], bins=10, edgecolor='black')

plt.title("Age Distribution")

plt.xlabel("Age (Normalized)")

plt.ylabel("Frequency")

plt.show()



# Scatter plot of Age vs Annual Income

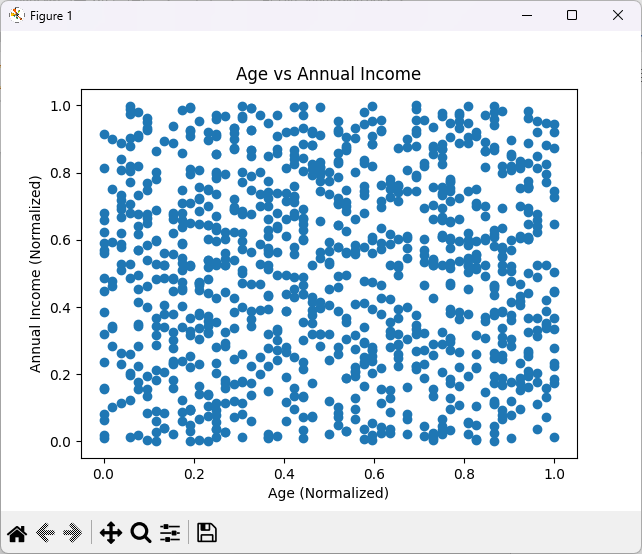
plt.scatter(customersData['Age'], customersData['AnnualIncome'])

plt.title("Age vs Annual Income")

plt.xlabel("Age (Normalized)")

plt.ylabel("Annual Income (Normalized)")

plt.show()



## Task 7:

## correlationMatrix = customersData[['Age', 'AnnualIncome', 'Purchased']].corr()

## print("\nCorrelation Matrix:")

## print(correlationMatrix)

## Task 8:

## customersData['IncomePerAge'] = customersData['AnnualIncome'] / customersData['Age']

## Task 9:

## customersData.drop(columns=['CustomerID'], inplace=True)

## # Split data into training and testing sets (80% training, 20% testing)

## X = customersData.drop(columns=['Purchased'])

## y = customersData['Purchased']

## X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Case Study 2: Employee Performance Prediction

## Task 1:

## count = 1500

## employeeID = np.arange(1, count + 1)

## age = np.random.randint(22, 61, count)

## yearsOfExperience = np.random.randint(1, 41, count)

## gender = np.random.choice(['Male', 'Female'], count)

## performanceRating = np.random.randint(1, 6, count)

## employeesData = pd.DataFrame({

## 'Employee ID': employeeID,

## 'Age': age,

## 'Years of Experience': yearsOfExperience,

## 'Gender': gender,

## 'Performance Rating': performanceRating

## })

## Task 2:

first15Rows = employeesData.head(15)

missingValues = employeesData.isnull().sum()

## Task 3:

medianIncome = employeesData['Years of Experience'].median()employeesData['Years of Experience'] = employeesData['Years of Experience'].fillna(medianIncome)

## Task 4:

labelEncoderGender = LabelEncoder()

employeesData['Gender'] = labelEncoderGender.fit\_transform(employeesData['Gender'])

## Task 5:

# Calculate Z-scores

z\_scores = np.abs(stats.zscore(employeesData['YearsOfExperience']))

# Identify outliers using a Z-score threshold (commonly set at 3)

threshold\_z = 3

outliers\_z = employeesData[z\_scores > threshold\_z]

# Remove Outliers

employeesData\_cleaned\_z = employeesData[z\_scores <= threshold\_z]

## Task 6:

scaler = MinMaxScaler()

employeesData[['Age', 'Years of Experience']] = scaler.fit\_transform(employeesData[['Age', 'Years of Experience']])

## Task 7:

# Boxplot of Performance Rating

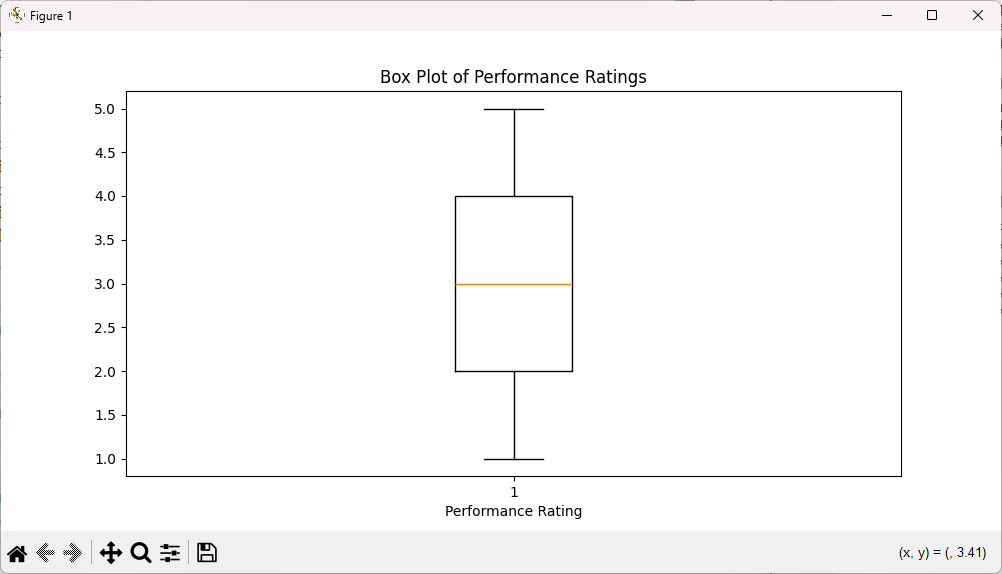
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.boxplot(employeesData['Performance Rating'], vert=False)

plt.title('Box Plot of Performance Ratings')

plt.xlabel('Performance Rating')



# Scatter plot of Age vs Annual Income

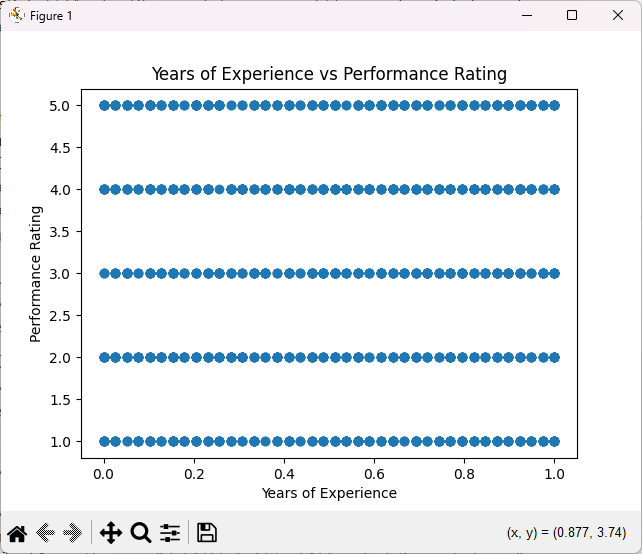
plt.scatter(employeesData['Years of Experience'], employeesData['Performance Rating'])

plt.title('Years of Experience vs Performance Rating')

plt.xlabel('Years of Experience')

plt.ylabel('Performance Rating')

plt.show()



## Task 8:

## correlationMatrix = employeesData[['Age', 'Years of Experience', 'Performance Rating']].corr()

## print("\nCorrelation Matrix:")

## print(correlationMatrix)

## Task 9:

## employeesData['IncomePerAge'] = employeesData['Years of Experience'] / employeesData['Age']

## Task 10:

## employeesData.drop(columns=['employeeID'], inplace=True)

## # Split data into training and testing sets (80% training, 20% testing)

## X = employeesData.drop(columns=['Performance Rating'])

## y = employeesData['Performance Rating']

## X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)